Comment on integration tests

```
A nice way to generate an integration test (like myhw2.main.TEST1):

+ write a driver with some print statements, like this:

    System.out.println(Data.inventory().get(v1));

  which may output something like

    Title1 (2000) : Director1 [10,0,0]

+ read the output to check it and make sure it's okay

+ then replace the System.output with an assert:

    Assert.assertEquals("Title1 (2000) : Director1 [10,0,0]",
                        Data.inventory().get(v1).toString());

Now you can run the test over and over and it will make sure that the
output is always what you got the time you checked it.
```

Comment on implementing iterators

```
For InventorySet.iterator(), it's easy just return
Collections.unmodifiableCollection(_data.values()).iterator();

For InventorySet.iterator(Comparator c), it is a bit more delicate.
Here's one way:

 + Create a separate copy in a list, for example
   List l = new ArrayList(_data.values())
 + Sort the list using Collections.sort(List,Comparator)
 + Return an iterator over the sorted list
```

Comment on testing iterators

```
How to test
InventorySet.iterator(), where we don't know the order...

 + Create a local HashSet<Video> and an InventorySet.
 + Put some videos in the InventorySet.
 + Put corresponding Videos in the HashSet.  Use Video, not Record,
   since you cannot create records directly.  You could also use
   strings such as "Title1 (2000) : Director1 [10,0,0]"
 + Now call InventorySet.iterator() and go through the elements.
   For each record returned do the following:
     ++ Make sure an equivalent video is in the local HashSet
     ++ Remove the equivalent video from the HashSet
 + Once you're done with the iterator, make sure the local HashSet is
   empty.

To test InventorySet.iterator(), where we do know the order ...

 + Create class that implements Comparator.  When called, the
   arguments will be Records.  For example, the following class will
   compare Records based on the number owned:

   Class NumOwnedComparator implements java.util.Comparator {
```

```
    public int compare (Object o1, Object o2) {
      Record r1 = (Record)o1;
      Record r2 = (Record)o2;
      return r2.numOwned() - r1.numOwned();
    }
  }

  or use the following in Java 1.5 to get rid of "unchecked
  conversion" warnings:

  class NumOwnedComparator implements java.util.Comparator<Record> {
    public int compare (Record r1, Record r2) {
      return r2.numOwned() - r1.numOwned();
    }
  }

+ Continue as before, but this time use a List to keep the videos in
  expected order.

If you want some more reading about Comparators/Comparable see:
Making Java Objects Comparable
by Budi Kurniawan
http://www.onjava.com/pub/a/onjava/2003/03/12/java_comp.html
```

Comment on the use of Command

```
Here is one scenario --- running an add command --- from line 31 of
myhw2.main.TEST1:

00029:     Video v1 = Data.newVideo("Title1", 2000, "Director1");
00031:     Assert.assertTrue(Data.newAddCmd(_inventory, v1, 5).run());

The objects involved are:

  +-----------+  +----------------+  +------------------+  +--------------+
  | t : TEST1 |  | c : CmdAdd     |  | i : InventorySet |  | v : VideoObj |
  | --------- |  | ----------     |  | ---------------- |  | ------------ |
  +-----------+  +----------------+  +------------------+  +--------------+
                 | inventory = i  |
                 | video = v      |
                 | change = 5     |
                 +----------------+

Execution proceeds as follows:

  t calls c.run();
  c calls i.addNumOwned(v,5)
```